Randy's SAS hints, updated Feb 6, 2014

1. Always begin your programs with internal documentation.

```
*  *****************
* Program =test1, Randy Ellis, first version: March 8, 2013
***************;
```

2. Don't forget semicolons and RUN statements The two most common programming errors.
   Also be careful about single and double quotes since they must be balanced.

3. Almost always use system options

```
options compress =yes  nocenter;
/* mostly use */
options ps=9999 ls=200;

*other key options to consider;
Options obs = max   /* or obs=100, Max= no limit on maximum number of obs processed */
Nodate nonumber /* useful if you don't want SAS to embed headers at top of each page in listing */
Macrogen     /* show the SAS code generated after running the Macros. */
Mprint   /* show how macro code and macro variables resolve */
nosource /* suppress source code from long log */
nonotes    /* be careful, but can be used to suppress notes from  log for long macro loops */
;                    *remember to always end with a semicolon!;
```

4. Use these three key commands regularly

```
Proc Contents data=test; run;
Proc means data = test (obs=100000); run;
Proc print data = test (obs=10); run;

*Toogle between;
Options obs =10;
*options obs = max;
```

5. Understand temporary versus permanent files;
   Data one;   creates a work.one temporary dataset that disappears when SAS terminates;
   Data out.one; creates a permanent dataset in th3e out directory that remains even if SAS terminates;

   Define libraries (or directories):

```
Libname out "c:/data/marketscan/output";
Libname in "c:/data/marketscan/MSdata";
```

Output or data can be written into external files:
Filename textdata “c:/data/marketscan/textdata.txt”;

6. Use arrays abundantly. You can use different array names to reference the same set of variables.

   Array x {100} X001-X100;
   Array xx {100} X001-X100 y1 y2 index etc;
   Array xxx X001-X100;

7. Name variables in arrays using leading zeros to improve sort order of variables
   Array x {100} X001-X100;
   not
   Array x {100} X1-X100;

8. Learn Set versus Merge command (ignore Update which is for rare, specialized use)

   Data three;   *information on the same person combined into a single record;
   Merge ONE TWO;
   BY IDNO;
   Run;

9. Learn key dataset options like
   Obs=
   Keep=
   Drop=
   In=
   Firstobs=
   Rename=(oldname=newname)
   End=

10. Keep files being sorted “skinny” by using drop or keep statements
    Proc sort data = IN.BIG(keep=IDNO STATE COUNTY FROMDATE) out=out.bigsorted;
    BY STATE COUNTY IDNO FROMDATE;
    Run;
    Also consider NODUP and NODUPKEY options to sort while dropping duplicate records, on all or on BY variables, respectively.

11. Take advantage of BY group processing
    Use FIRST.var and LAST.var abundantly.

    USE special variables

_N_ = current observation counter
ALL set of all variables such as Put _all_. Or when used with PROC CONTENTS, set of all datasets.
PROC CONTENTS data = in._all_; run;

12. Use lots of comments
    * this is a standard SAS comment that ends with a semicolon;
    /* a PL1 style comment can comment out multiple lines including comments;
        * Like this; */
    %macro junk; Macros can even comment out other macros or other pl1 style comments;
    /*such as this; */ * O Boy!; %macro ignoreme; mend; *very powerful;
    %mend; * end macro junk;

13. Use meaningful file names!
    Data ONE TWO THREE can be useful.

14. Put internal documentation about what the program does, who did it and when.
15. Learn basic macro language; See SAS program demo for examples. Know the difference between **executable** and **declarative** statements used in DATA step

    EXECUTABLE COMMANDS USED IN DATA STEP (Actually DO something, once for every record)

    Y=y+x (assignment. In STATA you would use GEN y=x or REPLACE Y=X)

    DO I = 1 TO 10;
    END; (always paired with DO, can be nested nearly unlimited deepness)

    INFILE in 'c:/data/MSDATA/claimsdata.txt'; define where input statements read from;
    FILE out 'c:/data/MSDATA/mergeddata.txt'; define where put statements write to;

    GOTO JOHNNY avoid always)

    IF a=b THEN y=0 ;
    ELSE y=x;
    CALL *subroutine(); (Subroutines are OK, Macros are better)*

    INPUT X ; (read in one line of X as text data from INFILE)
    PUT x y= / z date.; (Write out results to current LOG or FILE file)

    MERGE IN.A IN.B ; BY IDNO; * Match up with BY variable IDNO as you simultaneously read in A&B;

Both files must already be sorted by IDNO.

```
SET A B;                          * read in order, first all of A, and then all of B;
UPDATE   A B;  *replace variables with new values from B if non missing;

OUTPUT out.A;      *Write out one obs  to out.A SAS dataset;
OUTPUT;            *Writes out one obs of every output file being created;

STOP;                    * ends the current SAS datastep;
```

Assignment commands for DATA Step are
only done once at the start of the data step

```
DATA ONE TWO IN.THREE;
*This would create three data sets, named ONE TWO and IN.THREE
Only the third one will be kept once SAS terminates.;
Array x {10} x01-x10;
ATTRIB x length =16 Abc length=$8;
RETAIN COUNT 0;
BY state county IDNO;
   Also consider   BY DESCENDING IDNO; or BY IDNO UNSORTED; if grouped but not sorted by IDNO;
DROP i;    * do not keep i in final data set, although it can still be used while the data step is running
KEEP IDNO AGE SEX; *this will drop all variables from output file except these three;
FORMAT x date.;   *permanently link the format DATE. To the variable link;
INFORMAT ABC $4.;
LABEL AGE2010 = "Age on December 31 2010";
LENGTH x 8;  *must be assigned the first time you reference the variable;
RENAME  AGE = AGE2010;  After this point you must use the newname (AGE2010);
OPTIONS NOBS=100;  One of many options. Note done only once.
```

Key Systems language commands

```
LIBNAME to define libraries
FILENAME to define specific files, such as for text  data to  input our output
OPTIONS
TITLE THIS TITLE WILL APPER ON ALL OUTPUT IN LISTING;
%INCLUDE
%LET year=2011;
%LET ABC = "Randy Ellis";
```

Major procs you will want to master

```
DATA step  !!!!! Counts as a procedure;
Proc contents
```

PROC PRINT
PROC MEANS
PROC SORT
PROC FREQ frequencies
PROC SUMMARY (Can be done using MEANS, but easier)
PROC CORR (Can be done using Means or Summary)
PROC REG OLS or GLS
PROC GLM General Linear Models with automatically created fixed effects
PROC FORMAT /INFORMAT
PROC UNIVARIATE
PROC GENMOD nonlinear models
PROG SURVEYREG clustered errors

Formats are very powerful. Here is an example from the MarketScan data. One use is to simply recode variables so that richer labels are possible.

Another use is to look up or merge on other information in large files.

```
Proc format;
   value $region
      1='1-Northeast Region      '
      2='2-North Central Region    '
      3='3-South Region          '
      4='4-West Region           '
      5='5-Unknown Region        '
      ;

   value $sex
      1='1-Male       '
      2='2-Female     '
      other=' Missing/Unknown'
     ;

*Three different uses of formats;
Data one ;
sex='1';
region=1;
run;

data two;
set one;
```

Format sex $sex.; * permanently assigns sex format to this variable and stores format with the dataset;
Label sex = 'patient sex =1 if male';
Run;
Proc print data = two;
Run;
Proc contents data = two;
Run;
*be careful if the format is very long!;

Data three;
Set one;
Charsex=put(sex,$sex.);
Run;
*maps sex into the label, and saves a new variable as the text strings. Be careful can be very long;
Proc print data =three;
Run;

Proc print data = one;
Format sex $sex.;
Run;